AMENDMENTS TO THE SPECIFICATION

1. Please amend paragraph **[0047]** as follows:

[0047] In the above-described structure of the standardized command control table 34, a standardized command "typedef enum" such as "DIA_ COMMAND_ID_SDH_E" exists as "DIA_COMMAND_ID_PDH_E", "DIA_COMMAND_ID_ATM_E", *etc.* on a field basis. Various command functions can be arbitrarily added to the standardized command control table 34. The user of the higher-order application can arbitrarily call a corresponding device driver through the standardized command control table 34. Although a device driver is changed, a corresponding standardized command remains as it is. Only a function pointer is varied when the corresponding device driver carries out a corresponding command. The function pointer is provided to the DIA hierarchy 22 by the device driver when a device is initialized. The command is called through an API "Dia_Control" by the user of the higher-order application. When the user of the higher-order application calls a specific control command ~~the API "Dia_Control"~~ through the API "Dia_Control" ~~a specific control command~~, only a device driver positioned below the DIA hierarchy 22 is varied.

2. Please amend paragraph **[0050]** as follows:

[0050] An event table 38 shown in Fig. 5 is a structure located at a position pointed by the pointer of "*pEventTable" of a corresponding DCB being generated only if necessary. When the event table is not used, it is indicated as a null. The event table 38 includes an event ID of "EventId" and an event list structure pointer of "*pEventList". Each event list is a linked list as shown in Fig. 9. Accordingly, a plurality of places can be notified of one event and a call-back function of "Call-back Fn" can be called. The event table 38 initially includes the event ID of "EventId" and the pointer of "*pEventList" is initially pointed as a null. When the user of the higher-order application 20 uses a certain ~~corresponding~~ event, the user of the higher-order application 20 is able to register and use the event by connecting to "*pEventList", ~~can be connected to~~ a pointer of an event list structure using an API "Dia_Register".

AMENDMENTS TO THE CLAIMS

Please amend claim 18 as follows:

1. (Original) A method for commonly controlling device drivers, comprising the steps of:

arranging a device independent access hierarchy between an application hierarchy and a device driver hierarchy and applying a standardized rule of said device independent access hierarchy to said application hierarchy and said device driver hierarchy; and

allowing said application hierarchy and said device driver hierarchy to access the device driver hierarchy and said application hierarchy through the standardized rule of said device independent access hierarchy, respectively.

2. (Original) The method as set forth in claim 1, with said step of allowing said application hierarchy and said device driver hierarchy to access, comprising the steps of:

allowing said application hierarchy to transmit control commands based on a standardized common format for a corresponding device driver to said device independent access hierarchy, and allowing said device independent access hierarchy to convert the control commands into other control commands based on a local format and transmit the converted control commands to said device driver; and

allowing said device driver to give a response to the converted control commands based on the local format to said device independent access hierarchy, and allowing the device independent

access hierarchy to convert the response from said device driver into a response based on the standardized common format and transmit the response based on the standardized common format to said application hierarchy.

3. (Original) A method for commonly controlling device drivers, comprising the steps of:

arranging a device independent access hierarchy between an application hierarchy and a device driver hierarchy;

defining functions available in a corresponding device driver among functions of a function block in a function table;

when a device is initialized, allowing said device independent access hierarchy to generate a device handler identifier based on a standardized data format for said device and transmit the generated device handler identifier to the application hierarchy of a higher order; and

allowing the higher-order application hierarchy to call a predetermined device using the device handler identifier, and allowing said device independent access hierarchy to identify a function of the corresponding device driver from the function table using the device handler identifier and call the function of the corresponding device driver.

4. (Original) The method as set forth in claim 3, with said device handler identifier being represented as DCB handlerId[x1.x2.x3], where x1, x2 or x3 is an unsigned integer, x1 being a value of the level 1 meaning a device ID, x2 being a value of the level 2 meaning a logical or physical group number of a corresponding device, x3 being a value of a channel meaning a channel number

of a corresponding device or group.

5. (Original) The method as set forth in claim 4, with values of x1, x2 and x3 being "0" corresponding to there being no corresponding level or channel and the value of x1 sequentially increasing from "1"when the device is initialized.

6. (Original) A method for commonly controlling device drivers, comprising the steps of:

arranging a device independent access hierarchy between an application hierarchy and a device driver hierarchy;

when a device initialization is controlled by said application hierarchy, allowing said device independent access hierarchy to carry out level 1 initialization, level 2 initialization and channel initialization and generate a device handler identifier based on a standardized data format for a device;

allowing said device independent access hierarchy to dynamically assign a device control block, containing elements for carrying out a standardized rule, corresponding to said device handler identifier;

allowing said device independent access hierarchy to provide said device handler identifier to said application hierarchy; and

allowing said application hierarchy to call a predetermined device through said device independent access hierarchy using said device handler identifier.

7. (Original) The method as set forth in claim 6, with the elements of said device control block comprising a pointer of "*pControlTable" for pointing a position of a command control table, the command control table containing a command identifier having a standardized unique value and a command function pointer mapped to the command identifier, a pointer of "*pDDCB" for pointing a position of a device driver control table through which the existence and position of a corresponding function is identified, and a pointer "*pAnchor" for pointing a next level.

8. (Original) The method as set forth in claim 6, with the elements of said device control block comprising a pointer of "*pHandler" for pointing a position of a given initialization profile when a device is initialized, a function pointer of "*fpInitDevice" being used when a device is initialized, a function pointer of "*fpOpenChannel" being used when a channel is open, a function pointer of "*fpCloseChannel" being used when a channel is closed, a function pointer of "*fpRead" being used when data of an open channel is read, a function pointer of "*fpWrite" being used when data of the open channel is written, a function pointer of "*fpReset" being used when a device is reset, a pointer of "*pControlTable" for pointing a position of a command control table containing a command identifier having a standardized unique value and a command function pointer mapped to the command identifier, a pointer of "*pDDCB" for pointing a position of a device driver control table through which the existence and position of a corresponding function is identified, a pointer of "*pEventTable" for pointing a position of an event table, and a pointer "*pAnchor" for pointing a next level.

9. (Original) The method as set forth in claim 6, with the level 1 initialization of said device being made by giving a device identifier value of x1 as a unique value for each device based on a sequence of the level 1 initialization in the device handler identifier represented as DCB handlerId[x1.x2.x3] where x1, x2 or x3 is an unsigned integer.

10. (Original) The method as set forth in claim 9, with the level 2 initialization of the device being made by referring to the number of logical or physical groups, assigning anchors, and giving a group value of x2 as a unique value for each anchor in the device handler identifier represented as DCB handlerId[x1.x2.x3] where x1, x2 or x3 is an unsigned integer.

11. (Original) The method as set forth in claim 10, with the level 3 initialization of the device being made by giving a channel value of x3 for each of channels belonging to the device and groups within the device on the basis of an open channel sequence in the device handler identifier represented as DCB handlerId[x1.x2.x3] where x1, x2 or x3 is an unsigned integer.

12. (Original) A method, comprising:

requesting loss of signal state information based on a standardized common format by an application to a device independent access hierarchy;

converting the request from said application into a first device local format and requesting a first device driver to provide the loss of signal state information to said device independent access hierarchy;

responding to the request for loss of signal state information based on the first device local format;

responding to said application by said device independent access hierarchy for loss of signal state information based on the standardized common format.

13. (Original) The method of claim 12, with said step of converting the request from said application further comprising of converting the request into a second device local format and requesting a second device driver to provide the loss of signal state information to said device independent access hierarchy based on the second device local format when a first device is converted to a second device and said first device driver is changed to said second device driver.

14. (Original) The method of claim 13, further comprising of converting control commands based on the standardized common format to control commands provided to the device drivers accommodating a change of said application to a second application without changing the control commands provided to the device drivers.

15. (Original) The method of claim 14, further comprised of providing a mutual interface between said application and said first and second device drivers by the device independent access hierarchy reading material from a device driver control block and accessing the first and second device drivers using predetermined functions.

16. (Original) The method of claim 15, further comprising of said device independent access hierarchy using device handler identifiers based on the standardized data format, said device handler identifiers corresponding to respective devices.

17. (Original) The method of claim 16, further comprising:

providing the device handler identifiers to said application from said device independent access hierarchy during an initialization of the corresponding device; and

storing, by said application, the device handler identifiers and calling a corresponding device using a corresponding device handler identifier.

18. (Currently Once Amended) The method of claim 17, further comprising of said device independent access hierarchy determining according to said device handler identifier whether a certain device driver should be called and calling the certain driver handler ~~device driver~~ according to the determination.

19. (Original) The method of claim 18, with the device independent access hierarchy using certain pointers and function pointers in performing the standardized common format in the device independent access hierarchy.

20. (Original) The method of claim 19, further comprised of when said application is calling a function of a function block to be used, said device independent access hierarchy identifies the

existence of a corresponding function from a function table and uses a device handler identifier to inform the initialization of the device driver accommodating said application to access a device driver using said device handler identifier.

21. (Original) The method of claim 20, further comprised of not varying the device handler identifier value for the device when said first device driver is changed to said second device driver.

22. (Original) The method of claim 21, further comprising of varying the addresses of the pointers under the control of said device independent access hierarchy when said first device driver is changed to said second device driver.

REMARKS

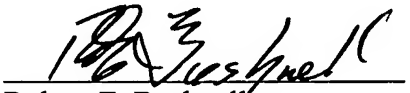
Claims 1 through 22 are pending in this application. Claim 18 is amended in several particulars for purposes of clarity in accordance with current Office policy, to assist the Examiner and to expedite compact prosecution of this application.

The specification is amended by this Second Preliminary Amendment to correct minor errors. No new matter is added.

In view of the foregoing Second Preliminary Amendment, this application is believed to be in condition for examination. Should questions arise during examination, the Examiner is requested to contact applicant's attorney.

No fee is incurred by this Second Preliminary Amendment. Should there be a deficiency in payment, or should other fees be incurred, the Commissioner is authorized to charge Deposit Account No. 02-4943 of Applicant's undersigned attorney in the amount of such fees.

Respectfully submitted,



Robert E. Bushnell,
Attorney for the Applicant
Registration No. 27,774

1522 "K" Street, N.W., Suite 300
Washington, D.C. 20005
(202) 408-9040

Folio: P56833
Date: 10/15/03
I.D.: REB/SS